**10/16/2017** **CSE 565: Computer Security** **Due: 11/14/2017**

# Project 2 – Block Ciphers: Symmetric Key and Asymmetric Key Crypto Systems

While Project 1 covered web security and program security, Project 2 (this project) closely follows the material covered in the class and it gives you an opportunity to put the various cryptographic principles into practice. In this project, you will learn more on encryption/decryption, cryptanalysis, and usage of cryptographic toolkits. This project is in two parts. In Part I, you will explore the operation of the AES encryption algorithm by tracing its execution, computing one round by hand, and then exploring the various block cipher modes of use. This is taken directly from the textbook author's site and is a routine exercise aimed at enhancing the understanding of AES and its use. Part II requires you to do some research, deep analysis, coding and use of the Sage package (ref. textbook) to handle complex computer algebra. For Part II, you may find the research paper listed at the end of this document quite useful. You may divide the work among yourselves within the groups, but all of you should be responsible for the entire project. Random groups may be asked to explain your solution at the time of grading the project reports.

## Part I – AES Encryption Algorithm

This project explores the operation of the AES encryption algorithm by tracing its execution, computing one round by hand, and then exploring the various block cipher modes of use. An online Java applet is used (or can be downloaded) to execute AES. The project is divided into three separate parts:

- **Block cipher internals:** This part involves encrypting plaintext and analyzing the intermediate results after each round. There is an online calculator for both AES and DES that provides the intermediate results and the final ciphertext.
- **Block cipher round:** This part involves calculating one round by hand and comparing the results to those produced by the calculator.
- **Block cipher modes of use:** Enables the student to compare the operation of CBC and CFB modes.

Use the link: http://williamstallings.com/Crypto/AESCalc/AESlab.html to go over the project guidelines, deliverables and what is expected in the project report. **Group leaders must send an email to** shambhu@buffalo.edu **to receive the AES Triple that is needed to complete this project.**

## Part II – Cryptanalysis and OPENSSL

### 1. <u>Background</u>

The specific objective of this part of the project is to first understand the working of an encryption algorithm, analyze the protocol to view a flaw and then try to exploit it. This project is developed along the fault-based attack described in Section 9.2 of the textbook. However, it should be noted that the protocol might not have a flaw that can be easily exploited.

Read the entire project description carefully to understand what is expected from you in this part of the project.

### 2. <u>Prerequisites</u>

In the first phase of this part, our objective is to set up a virtual Linux machine to help you with the project. If you have already set up a virtual machine on your laptop, you may skip this step. You will then install the OpenSSL toolkit. We will use Ubuntu installed on a virtual machine for achieving this.

***<u>To install a virtual machine, follow the steps given in Project 1.</u>***

***<u>To install OpenSSL follow these steps:</u>***

Follow step (a)-(b) to download the latest OpenSSL package and its MD5 hash, and then verify the integrity of the downloaded package (If you need more help, you may refer to: http://www.linuxfromscratch.org/blfs/view/svn/postlfs/openssl.html or https://geeksww.com/tutorials/libraries/openssl/installation/installing_openssl_on_ubuntu_linux.php).

(a) Run the following commands as root (for privileges on destination directory), or else you can use "sudo" command; replace the xyz with the latest source tar-ball name. You can also download them manually from the OpenSSL website.
```
#sudo wget http://www.openssl.org/source/openssl- xyz.tar.gz
#sudo wget http://www.openssl.org/source/openssl- xyz.tar.gz.md5
```

(b) Run the following commands to verify the integrity of the downloaded package.
```
#md5sum openssl- xyz.tar.gz
#cat openssl- xyz.tar.gz.md5
```

The two commands in step (b) generate two strings of alpha-numeric characters. Check to see if both strings are identical or not. If not, repeat step (a) for a different version. If yes, your file has been downloaded properly. Note that if you are using openssl for a highly secure/critical setup (or for any other reason) then you should also check the PGP signatures (not covered here for simplicity).

Follow steps (c)-(g) to install OpenSSL. Check for any error message for these steps.

(c) Extracting files from the downloaded package.
```
#sudo gunzip openssl- xyz.tar.gz
#sudo tar -xvf openssl- xyz.tar
```
(d) Now, enter the directory where the package is extracted.
```
#cd openssl- xyz
```
(e) Configuring OpenSSL.
```
#sudo ./config --prefix=/usr/local/openssl --openssldir=/usr/local/openssl
```
Replace "/usr/local/openssl" above with the directory path where you want to copy the files and folders.

(f) Compiling OpenSSL.
```
#sudo make
```
(g) Installing OpenSSL.
```
#sudo make install
```

## 3. Operational Details

We will be using OpenSSL to understand the basics of an encryption/decryption algorithm. The documentation for OpenSSL can be found at http://www.openssl.org/docs/.

### Phase 1: *A decryption protocol:*

In this phase, we will be using a decryption protocol that uses asymmetric cryptography (PKI). A user has a Public Key (N,e) that is published on a server. The ciphertext is created via a RSA-style encryption process.

$C = M^e \bmod N$ where $N = p \times q$ and p and q are two very large primes.

The decryption process however works on a different basis.

1. To get back the plaintext one uses:
$$M_1 = {C_p}^{\alpha} \bmod p$$
$$M_2 = {C_q}^{\beta} \bmod q$$
**where** $\alpha = d \bmod (p-1)$ ; $\beta = d \bmod (q-1)$ ; $C_p = C \bmod p$ and $C_q = C \bmod q$
2. **Calculate** two constants: $A = q^{p-1} \bmod N$ and $B = p^{q-1} \bmod N$
3. **Calculate** the coefficients: $SH_1 = M_1 A \bmod N$ and $SH_2 = M_2 B \bmod N$
4. The final message is got back as follows:
$M = SH_1 + SH_2$. However, if $M \geq N$ then $M = M - N$.

Such an algorithm is used in embedded devices for a variety of reasons (why?).

However, this process has a fatal vulnerability, where if a user somehow (by mistake, e.g.,) changes the method of calculating $M_2$ and instead calculates a $M_2'$ such that $M_2' \neq {C_q}^{\beta}\ mod\ q$, then an opponent can make use of the protocol to find the keys.

**<u>Phase 2</u>: *An alternative encryption protocol:***

Using the decryption rationale as explained above, one should be able to generate a ciphertext in embedded devices (why?).

The logic would be as follows:

**Calculate**

$$C_1 = M^{\alpha}\ mod\ p$$
$$C_2 = M^{\beta}\ mod\ q$$
$$C = S \times C_1 + T \times C_2\ mod\ n$$

**Note:** α *and* β *are different from Phase 1 and need to be redefined*

**where**

$$S \equiv 1\ mod\ p\ ; S \equiv 0\ mod\ q$$
$$T \equiv 1\ mod\ q ; T \equiv 0\ mod\ p$$

Would this protocol also show the same weakness? Justify your answer.

**<u>Phase 3</u>: *A case study:***

There is a text file that has been intercepted by an adversary who knows that this file has been encrypted by an AES cipher in ECB mode using a 128-bit key. The AES key however is encrypted using a PKI method that we think is RSA.

Provided to you are the following:

1. RSA Public key in a file named public.key
2. Encrypted AES key in misc.txt
3. Encrypted text file as encrypted.aes (not readable on your editor)

These files are downloadable from the Handouts page on the class website. Here is the link for the handouts page: http://www.cse.buffalo.edu/~shambhu/cse56517/handouts.html.

You also have the following information:

1. The embedded device has been tested and we have realized that the same key can be used to generate signatures as well as decryption. However, the generation of signature is done using the alternative encryption protocol.
2. We have generated two signatures over the same data and we hope to exploit the vulnerability of the alternative encryption protocol. Therefore you have two ciphertexts, C1 and C2 out of which C1 is correct, whereas C2 is tampered (see NOTE below).

***<u>NOTE</u>***: $C1 \neq C_1\ and\ C2 \neq C_2$. but $C1 = f(C_1, C_2)$ whereas $C2 = f(C_1, C_2')$.

You have to ***<u>try</u>*** to get the private key and the contents of the AES encrypted file.

## 4. <u>What You are Expected to Learn</u>

You are expected to learn

(a) How to set up a virtual box and install Ubuntu (this part is not included in the grading)

(b) Installing OpenSSL

(c) Understanding the math behind the decryption algorithm as well as the alternative encryption algorithm.
(d) Understanding vulnerabilities of algorithms
(e) Using OpenSSL encryption and decryption algorithms (using AES and RSA)

## 5. Project Deliverables

Designate this as Part II of your project report with the following sections:

**Section 1:** Describe the reasons for having the decryption process as explained in Sec. 3, Phase 1 for embedded devices.
**Section 2:** Explain in detail your analysis about the vulnerability of the decryption process. Your answer should be clear and concise.
**Section 3**: Describe the reasons for having the encryption process as explained in Sec. 3, Phase 2 for embedded devices. Would this protocol also show the same weakness? Justify your answer.
**Section 4:** Provide in detail your steps and analysis of Sec. 3, Phase 3. You can use OpenSSL to try to decrypt files or get additional information. Include screen shots with the decryption command you execute and the results you get if any.

Submit a project report with two parts, viz. Part I for the AES project and Part II for the OpenSSL project, no more than 8-10 pages total, clearly with the name (lastname, firstname) and person numbers of all group members and indicating their contributions. Include all the above sections and clearly demarcate each section under a separate heading.

## 6. Project Guidelines (Important)

For the projects, you continue to work in groups. If necessary, rotate the role of group leader from the previous project. Having a leader helps in the completion of the project in a timely manner. A single report per group is desired. If you refer to any particular paper or web document for this project, remember to **mention the source in your report**. Otherwise, such actions will be construed as plagiarism and will **NOT** be graded and appropriate penalties will be applied.

### Operating Systems Allowed

**1)** Windows 7; **2)** Ubuntu

### Additional Guidelines

- A major part of this project is to learn how to debug and install the software. Since every system is unique, we will not be able to help you with installation problems.
- Some additional software like MATLAB or SAGE might be useful for the project. The CSE machines have MATLAB installed on them and SAGE can be downloaded from http://www.sagemath.org/.
- You will be graded based *only* on what you have written in your report.

## 7. References

Dan Boneh, Richard A. DeMillo, Richard J. Lipton: On the Importance of Eliminating Errors in Cryptographic Computations. J. Cryptology 14(2): 101-119 (2001).